

This is not the final version yet!

What is your unit test coverage:

4 sets of test packages : Cardmatching, gamecentre, sliding tiles, sudoku.

Cardmatching test package : 1. CardMatchingBoardManagerTest class

2. CardMatchingBoardTest class

CardMatchingBoardManagerTest covers the line % as follows:

71% lines covered in Card class

86% lines covered in CardMatchingBoard class

73% lines covered in CardMatchingBoardManager class

CardMatchingBoardTest covers the line %:

66% lines covered in Card class

45% lines covered in CardMatchingBoard class

Gamecentre test package : 1. GameCentreTest class 2.SavedGamesManagerTest class

3. ScoreBoardTest class 4. ScoreTest class

GameCentreTest class line percent coverage:

42% lines covered in GameCentre class

5 % lines covered in SavedGamesManager class

4 % lines covered in ScoreBoard class

10% lines covered in UserManager class

SavedGamesManagerTest class line percent coverage:

100% lines covered in GameToSave class

98% lines covered in SavedGamesManager class

22% lines covered in User class

ScoreBoardTest class line percent coverage:

92% lines covered in Score class

95% lines covered in ScoreBoard class

ScoreTest class line percent coverage:

100% lines covered in Score class

Sliding tiles test package: 1. SlidingTilesBoardManagerTest class 2.

SlidingTileBoardTest class

SlidingTilesBoardManagerTest class line percent coverage:

69% lines covered in SlidingTileBoard class

100% lines covered in SlidingTileBoardManager class

94% lines covered in Tile class

SlidingTilesBoardTest class line percent coverage:

94% lines covered in SlidingTileBoard class

44% lines covered in Tile class

Package Sudoku test package :1 .sudoku board manager test 2. Sudoku board test 3. Sudoku play board test

Sudoku board manager test :

97% lines covered for SudokuPlayBoard class

100% lines covered for SudokuBoardRandomizer class

100% lines covered for SudokuBoardManager class

96% lines covered for SudokuBoard class.

Sudoku board test: 98% lines covered for SudokuBoard class

Sudoku play board test:

94% lines covered for SudokuPlayBoard class

26% lines covered for SudokuBoardRandomizer class

94% lines covered for SudokuBoard class

What are the most important classes in our program:

Card matching package: Card matching board manager , card matching board

Gamecentre package: Game centre activity , game centre , game to save, saved games manager, score board, user manager

Sliding tiles package: Sliding tile board manager, sliding tile board

Sudoku package: Sudoku board, sudoku board manager, sudoku playboard

What design patterns did you use? What problems do each of them solve?

Dependency injection for sudokuBoard and SudokuPlayBoard class. Problems solved: it is more reusable and testable.

We followed the MVC design pattern for our games. This MVC design pattern separates the application into three main parts : model ,view and controller. So it allows us to implement each component independently and don't mess each other around.

How did you design your score board ? Where are high scores stored ? How did they get displayed?

There are two classes that are responsible for keeping track of user score. There is a Score class that is used to keep track of a single score. The Score class contains information about the numeric score, the user that achieved the score, and the game that the score is set in. The Score class also implemented Comparable<Score> for comparison between two scores.

Then there is the ScoreBoard class which keeps track of all the scores on the device with two hashmaps. There is only one instance of ScoreBoard, and is serialized + loaded by the class GameCentre every time it needs to be read/updated.

One hashmap (byUser) has the username of all the users that has logged in before as keys, and the other hashmap (byGame) has the game name as keys. Both of these hashmaps have ArrayList<Score> as values. They differ in that the values of the hashmap byUser is are lists of scores achieved by a user (which is the key for getting that value), while the values of hashmap byGame are lists of scores achieved in a game(the key for the value). ScoreBoard is implemented this way for reducing the method complexity needed to display scores in different ways(per game / per user/ all users in a game).

The ScoreBoard class only contains two public methods, updateScore() and createDisplayString. createDisplayString() is what is called when the score needs to be displayed in an Activity. createDisplayString() can generate 2 types of strings, a string containing only scores, and a string containing only gamename/username. It is implemented this way because HighScoreActivity has two different TextViews for the description of the score, and the score itself.

createDisplayString() takes in 3 parameters (String select, String username, boolean isScore). The select parameter is the way the score needs to be displayed (score of a user/ score of a game/ score of a user in a game / ...). The username parameter is the username of the current user that is logged in. Only scores related to the current user that is logged in will be displayed unless specified by the user to display all scores on the device. The third parameter isScore is used to distinguish which type of string needs to be created for display.

In HighscoreActivity, createDisplayString() is called twice, one for creating the description of the scores, and one for the list of numeric scores.